# A Piecewise Linear Dual Phase-1 Algorithm for the Simplex Method with All Types of Variables

István Maros

Department of Computing, Imperial College, London

Email: i.maros@ic.ac.uk

Departmental Technical Report 2000/13

ISSN 1469–4174

**Abstract**

A dual phase-1 algorithm for the simplex method that handles all types of variables is presented. In each iteration it maximizes a piecewise linear function of dual infeasibilities in order to make the largest possible step towards dual feasibility with a selected outgoing variable. The new method can be viewed as a generalization of traditional phase-1 procedures. It is based on the multiple use of the expensively computed pivot row. By small amount of extra work per iteration, the progress it can make is equivalent to many iterations of the traditional method. In addition to this main achievement it has some further important and favorable features, namely, it is very efficient in coping with degeneracy and numerical difficulties. Both theoretical and computational issues are addressed. Examples are also given that demonstrate the power and flexibility of the method.

**Keywords:** Linear programming, Dual simplex method, Phase-1, Piecewise linear functions.

# 1 Introduction

The dual simplex algorithm (DSA) developed by Lemke [7] has long been known as a better alternative to Dantzig's primal simplex [2] for solving certain types of linear programming (LP) problems, mostly in cases where a dual feasible solution is readily available. In an earlier paper [9] we presented a piecewise linear dual phase-2 procedure that handles all types of variables and is especially efficient in the presence of many upper bounded variables. This latter situation is typical in the LP relaxation of mixed integer programming problems within branch and bound type solution algorithms. In such a case there are many upper bounded variables and an optimal basic solution to a node problem which is dual feasible for the immediate successor problems is available. However, in many other cases a dual feasible solution is not available and the dual algorithm cannot be used even if it would be advantageous.

There are some techniques to obtain a dual basic feasible solution for the problem. The most typical one is the dual variant of the big-M method. In this case an extra '$\leq$' type constraint $\sum x_j$ is added to the original constraints. It is made non-binding by setting the right-hand-side coefficient equal to a large number (the big M). This enables the gradual build-up of a dual feasible basis if one exists. For details c.f., [12]. This and the other methods have been worked out for the case when the LP problem is in the standard form. If all types of variables are present in a problem the situation is more complicated. In such a case, one possibility is to introduce additional variables and constraints to convert the problem into the standard form with nonnegative variables only and apply the big-M or other methods.

In this paper we propose a more efficient algorithm to obtain a dual feasible solution for a problem with all types of variables. Our motivation was the computational enhancement of dual phase-1 in this general case. The key idea in the new approach is the multiple use of the (expensively) updated pivot row. The algorithm is a modification of the DSA such that in each iteration the largest possible step is made with a selected outgoing variable towards dual feasibility. This is achieved by maximizing a concave piecewise

linear function in every iteration. We show that the extra work per iteration is little and the algorithm can result in a considerably increased efficiency by greatly reducing the number of iterations in dual phase-1. This algorithm is monotone in the sum of dual infeasibilities. As such, the number of infeasibilities can even increase (though in this case the sum will definitely decrease), remain the same (even in the nondegenerate case, though infeasible positions may change) or decrease. The algorithm has some inherent flexibility that can alleviate or overcome occasional numerical and/or algorithmic (like degeneracy) difficulties.

The rest of the paper is organized as follows. In section 2 we state the primal and dual problems with all types of variables and discuss relevant work done in this area. Section 3 gives an analysis of dual infeasibility, it introduces a new dual phase-1 procedure and gives its algorithmic description. Section 4 presents two numerical examples that illustrate the operation of the procedure. It is followed by a summary and conclusions in section 5.

# 2 Problem statement

## 2.1 The primal problem

We consider the following primal linear programming (LP) problem:

$$\begin{aligned} \text{minimize} \quad & c^T x, \\ \text{subject to} \quad & Ax = b, \\ & l \leq x \leq u, \end{aligned}$$

where $A \in \mathbb{R}^{m \times n}$, $c$, $x$, $l$ and $u \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$. Some or all of the components of $l$ and $u$ can be $-\infty$ or $+\infty$, respectively. $A$ itself is assumed to contain a unit matrix $I$, that is, $A = [I, \bar{A}]$, so it is of full row rank. Variables which multiply columns of $I$ transform every constraint to an equation and are often referred to as *logical variables*. Variables which multiply columns of $\bar{A}$ are called *structural variables*.

By some elementary transformations it can be achieved that the variables (whether logical or structural) fall into four categories as shown in Table 1 (for further details, see

Orchard-Hays [11]).

| Feasibility range | Type | Reference |
|---|---|---|
| $x_j = 0$ | 0 | Fixed variable |
| $0 \leq x_j \leq u_j < +\infty$ | 1 | Bounded variable |
| $0 \leq x_j \leq +\infty$ | 2 | Non-negative variable |
| $-\infty \leq x_j \leq +\infty$ | 3 | Free variable |

Table 1: Types of variables

## 2.2 The dual problem

First, we restate the primal problem to contain bounded variables only.

$$\begin{aligned} (P1) \quad & \text{minimize} && c^T x, \\ & \text{subject to} && Ax = b, \\ & && 0 \leq x \leq u, \end{aligned}$$

where all components of $u$ are finite.

A basis to (P1) is denoted by $B$ and is assumed (without loss of generality) to be the first $m$ columns. Thus, $A$ is partitioned as $A = [B, N]$, with $N$ denoting the nonbasic part of $A$. The components of $x$ and $c$ are partitioned accordingly. Column $j$ of $A$ is denoted by $a_j$. A basic solution to (P1) is

$$x_B = B^{-1}\left(b - \sum_{j \in U} u_j a_j\right),$$

where $U$ is the index set of nonbasic variables at upper bound. The $i$th basic variable is denoted by $x_{Bi}$. The $d_j$ reduced cost of variable $j$ is defined as $d_j = c_j - \pi^T a_j = c_j - c_B^T B^{-1} a_j$ which is further equal to $c_j - c_B^T \alpha_j$ if the notation $\alpha_j = B^{-1} a_j$ is used.

The dual of (P1) is:

$$\begin{aligned} (D1) \quad & \text{maximize} && b^T y - u^T w, \\ & \text{subject to} && A^T y - w \leq c, \\ & && w \geq 0, \end{aligned}$$

where $y \in \mathbb{R}^m$ and $w \in \mathbb{R}^n$ are the dual variables. Stating the dual when the primal has all types of variables is cumbersome. However, we can think of the reduced costs of the primal problem as the logical variables of the dual [11]. In this way dual feasibility can be expressed quite simply as shown in the next section.

In practice, dual algorithms work on the primal problem using the computational tools of the sparse primal simplex method but perform basis changes according to the rules of the dual.

The upper bounded version of the DSA was first described by Orchard-Hays [11] and later by Chvátal [1]. They both deal with dual phase-2. Maros also published a general dual phase-2 algorithm (BSD in [9]) that handles all types of variables and appears to be computationally very efficient.

For phase-1 of the primal with all types of variables Maros developed the FEWPHI algorithm [8] that can make several iterations with one updated column. Similar ideas for the primal with slightly different scope are also discussed by Wolfe [13] and Greenberg [5].

For dual phase-1 the most relevant work is [4] in which Fourer gives a theoretical discussion of a procedure that is based on the above advanced ideas for the primal. His algorithm is monotone in both the sum and number of infeasibilities.

The creation of the updated pivot row $p$, i.e., the computation of $\alpha_{pj}$ for all nonbasic indices $j$ is an expensive operation in SSX (c.f. [10]). Traditional dual methods make *one iteration* with the pivot row and discard it. It will be shown that the new method, *GDPO (Generalized Dual Phase One)*, that takes advantage of some ideas of BSD and handles all types of variables, makes *one step* with the pivot row but it corresponds to (rarely) one or (frequently) many iterations of the traditional method at the price of one. GDPO is monotone only in the sum of infeasibilities which increases its flexibility. It also has some additional favorable features that enhance its effectiveness and efficiency. As a result, GDPO seems to be an attractive substitute for a general dual phase-1 algorithm.

Table 2: Dual feasibility of nonbasic $d_j$s (primal minimization)

| Type of nonbasic variable | Dual feasibility |
|---|---|
| 0 | $d_j$ of any sign |
| 1 | $d_j \geq 0$ if $x_j = 0$<br>$d_j \leq 0$ if $x_j = u_j$ |
| 2 | $d_j \geq 0$ |
| 3 | $d_j = 0$ |

# 3 Dual phase-1 with all types of variables

If there are only type-2 variables in the problem then $w$ is not present in (D1). Assuming that $B$ is a basis to $A$, dual feasibility is expressed by

$$B^T y = c_B$$
$$N^T y \leq c_N, \quad \text{or } d_N = c_N - N^T y \geq 0.$$

In practice, of course, all types of variables are present in a problem and it is desirable to handle them algorithmically rather than introducing additional variables and constraints and reverting to the traditional formulation. The $d_j$ of a basic variable is zero. For nonbasic variables the dual feasible values are shown in Table 2 (for proof c.f. [11]).

Since the $d_j$ of a type-0 variable is always feasible such variables can be, and in fact are, ignored in dual phase-1. Any $d_j$ that falls outside the feasibility range is dual infeasible. We define two index sets of dual infeasible variables:

$$M = \{j : (x_j = 0 \text{ and } d_j < 0)\}, \tag{1}$$

and

$$P = \{j : (x_j = u_j \text{ and } d_j > 0) \text{ or } (\text{type}(x_j) = 3 \text{ and } d_j > 0)\},$$

where type($x_j$) denotes the type of $x_j$.

There is an easy way to make the $d_j$ of upper bounded variables feasible. They can be infeasible in two different ways. Accordingly, we define two index sets:

$$T^+ = \{j : \text{ type}(x_j) = 1 \text{ and } j \in P\}$$
$$T^- = \{j : \text{ type}(x_j) = 1 \text{ and } j \in M\}$$

If we perform a bound swap for all such variables, the corresponding $d_j$s become feasible. In this case the basis remains unchanged but the solution has to be updated:

$$x_B := x_B - \sum_{j \in T^+} u_j \alpha_j + \sum_{j \in T^-} u_j \alpha_j, \tag{2}$$

where $\alpha_j = B^{-1} a_j$, ':=' denotes assignment, and the sum is defined to be 0 if the corresponding index set is empty. Computing $\alpha_j$ for all $j$ in (2) would be relatively expensive. However, this entire operation can be performed in one single step for all variables involved in the following way.

$$\begin{aligned} x_B &:= x_B - \sum_{j \in T^+} u_j \alpha_j + \sum_{j \in T^-} u_j \alpha_j \\ &= x_B - B^{-1} \left( \sum_{j \in T^+} u_j a_j - \sum_{j \in T^-} u_j a_j \right) \\ &= x_B - B^{-1} \tilde{a} \end{aligned} \tag{3}$$

with the obvious interpretation of $\tilde{a}$. Having constructed $\tilde{a}$, we need only one FTRAN operation with the inverse of the basis.

Assuming that such a *dual feasibility correction* has been carried out the definition of $P$ simplifies to:

$$P = \{j : \text{ type}(x_j) = 3 \text{ and } d_j > 0\}, \tag{4}$$

While this redefinition is not really necessary at this stage it will be very useful for the new algorithm.

Using infeasibility sets $M$ and $P$, the sum of dual infeasibilities is defined as

$$f = \sum_{j \in M} d_j - \sum_{j \in P} d_j, \tag{5}$$

where any of the sums is zero if the corresponding index set is empty. It is always true that $f \leq 0$. In dual phase-1 the objective is to maximize $f$ subject to the dual feasibility constraints. When $f = 0$ is reached the solution becomes dual feasible. If it cannot be achieved the dual is infeasible.

The dual simplex method performs basis changes using the computational tools of the primal. However, in dual the pivot row (the outgoing variable) is selected first, followed by a dual ratio test to determine the pivot column (incoming variable).

Let us assume row $p$ is selected somehow (i.e., the $p$th basic variable $x_{Bp}$ will leave the basis). The elimination step of the simplex transformation subtracts some multiple of row $p$ from $d_N$. If this multiplier is denoted by $t$ then the transformed value of each $d_j$ can be written as a function of $t$:

$$d_j^{(p)}(t) = d_j - t\alpha_{pj}. \tag{6}$$

With this notation, $d_j^{(p)}(0) = d_j$ and the sum of infeasibilities as a function of $t$ can be expressed (assuming $t$ is small enough such that $M$ and $P$ remain unchanged) as:

$$f^{(p)}(t) = \sum_{j \in M} d_j^{(p)}(t) - \sum_{j \in P} d_j^{(p)}(t) = f^{(p)}(0) - t \left( \sum_{j \in M} \alpha_{pj} - \sum_{j \in P} \alpha_{pj} \right).$$

Clearly, $f$ of (5) can be obtained as $f = f^{(p)}(0)$. To simplify notations, we drop the superscript from both $d_j^{(p)}(t)$ and $f^{(p)}(t)$ and will use $d_j(t)$ and $f(t)$ instead.

The change in the sum of dual infeasibilities, if $t$ moves away from 0, is:

$$\Delta f = f(t) - f(0) = -t \left( \sum_{j \in M} \alpha_{pj} - \sum_{j \in P} \alpha_{pj} \right). \tag{7}$$

Introducing notation

$$v_p = \sum_{j \in M} \alpha_{pj} - \sum_{j \in P} \alpha_{pj} \tag{8}$$

(7) can be written as $\Delta f = -tv_p$. Therefore, requesting an improvement in the sum of dual infeasibilities ($\Delta f > 0$) is equivalent to requesting

$$-tv_p > 0$$

which can be achieved in two ways:

$$\text{If } v_p > 0 \text{ then } t < 0 \text{ must hold,} \tag{9}$$

$$\text{if } v_p < 0 \text{ then } t > 0 \text{ must hold.} \tag{10}$$

As long as there is a $v_i \neq 0$ with type$(x_{Bi}) \neq 3$ (type-3 variables are not candidates to leave the basis) there is a chance to improve the dual objective function. The precise conditions will be worked out in the sequel. From among the candidates we can select $v_p$ using some simple or sophisticated (steepest edge type) rule.

Let $k$ denote the original index of the $p$th basic variable $x_{Bp}$, i.e., $x_k = x_{Bp}$ (which is selected to leave the basis). At this point we stipulate that after the basis change $d_k$ *of the outgoing variable take a feasible value.* It corresponds to releasing the $p$-th dual basic (equality) constraint so that it remains a satisfied inequality. This is not necessary but it gives a better control of dual infeasibilities.

If $t$ moves away from zero (increasing or decreasing as needed) some of the $d_j$s move toward zero (the boundary of their feasibility domain) either from the feasible or infeasible side and at a specific value of $t$ they reach it. Such values of $t$ are determined by:

$$t_j = \frac{d_j}{\alpha_{pj}}, \text{ for some nonbasic } j \text{ indices}$$

and they enable a basis change since $d_j(t)$ becomes zero at this value of $t$, see (6). It also means that the $j$-th dual constraint becomes tight at this point. Let us assume the incoming variable $x_q$ has been selected. Currently, $d_k$ of the outgoing basic variable is zero. After the basis change its new value is determined by the transformation formula of the simplex method giving

$$\bar{d}_k = -\frac{d_q}{\alpha_{pq}} = -t_q,$$

which we want to be dual feasible. The proper sign of $\bar{d}_k$ is determined by the way the outgoing variable leaves the basis. This immediately gives rules how an incoming variable can be determined once an outgoing variable (pivot row) has been chosen. Below is a verbal description of these rules. Details are given in the next section.

1. If $v_p > 0$ then $t_q < 0$ is needed for (9) which implies that the $p$th basic variable must leave the basis at lower bound (because $\bar{d}_k$ must be nonnegative for feasibility). In the absence of dual degeneracy this means that $d_q$ and $\alpha_{pq}$ must be of opposite sign. In other words, the potential pivot positions in the selected row are those that satisfy this requirement.

2. If $v_p < 0$ then $t_q > 0$ is needed which is only possible if the outgoing variable $x_{Bp}$ (alias $x_k$) is of type-1 leaving at upper bound. In the absence of degeneracy this means that $d_q$ and $\alpha_{pq}$ must be of the same sign.

3. If $v_p \neq 0$ and the outgoing variable is of type-0 then the sign of $d_q$ is immaterial. Therefore, if $v_p > 0$ we look for $t_q < 0$ and if $v_p < 0$ choose from the positive $t$ values.

It remains to see how vector $v = [v_1, \ldots, v_m]^T$ can be computed for row selection. In vector form, (8) can be written as

$$v = \sum_{j \in M} \alpha_j - \sum_{j \in P} \alpha_j = B^{-1} \left( \sum_{j \in M} a_j - \sum_{j \in P} a_j \right) = B^{-1} \tilde{a} \tag{11}$$

with obvious interpretation of auxiliary vector $\tilde{a}$. The latter is an inexpensive operation in terms of the revised simplex method.

## 3.1 Analysis of the dual infeasibility function

We can investigate how the sum of dual infeasibilities, $f(t)$, changes as $t$ moves away from 0 ($t \geq 0$ or $t \leq 0$). We show that, in either case, $f(t)$ is a piecewise linear concave function with break points corresponding to different choices of the entering variable. The global maximum of this function is achieved when its slope changes sign. It gives the maximum improvement in the sum of dual infeasibilities that can be achieved with the selected outgoing variable by making multiple use of the updated pivot row.

Let the index of the pivot row be denoted by $p$, the outgoing variable by $x_{Bp}$ ($\equiv x_k$) and the index of the incoming variable by $q$. The pivot element is $\alpha_{pq}$.

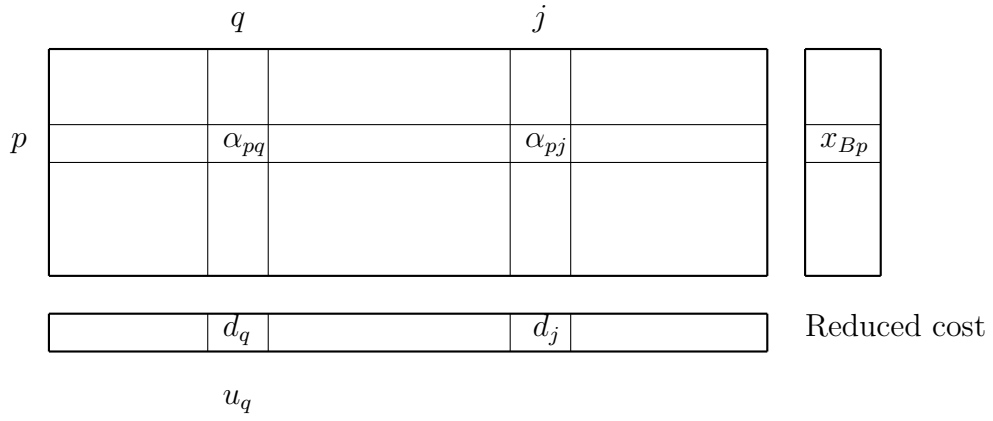


Figure 1: The key elements of the dual pivot step.

After the basis change, the new values of $d_j$ are determined by:

$$\bar{d_j} = d_j - \frac{d_q}{\alpha_{pq}}\alpha_{pj} \quad \text{for} \quad j \in N, \tag{12}$$

and for the leaving variable:

$$\bar{d_k} = -\frac{d_q}{\alpha_{pq}}.$$

The feasibility status of a $d_j$ (described in Table 2) may change as $t$ moves away from zero. We assume that feasibility correction for upper bounded variables has been carried out thus the corresponding $d_j$s are at feasible level. The following analysis uses (6) and (12) to keep track of the changes of the feasibility status of each $d_j(t)$.

**I.** $t \geq 0$, i.e., the outgoing variable leaves at upper bound.

(a) $\alpha_{pj} > 0$, $d_j(t)$ is decreasing

i. $d_j(0) > 0$

A. If $d_j(0)$ is infeasible, i.e., $j \in P$, type$(x_j) = 3$ (free), $d_j(t)$ remains infeasible as long as $t < \dfrac{d_j(0)}{\alpha_{pj}}$ and it becomes infeasible again if $t > \dfrac{d_j(0)}{\alpha_{pj}}$ when $j$ joins $M$.

B. If $d_j(0)$ is feasible, $d_j(t)$ remains feasible as long as $t \leq \dfrac{d_j(0)}{\alpha_{pj}}$ after which it becomes negative and $j$ joins $M$.

ii. If $d_j(0) = 0$ and $x_j$ is at lower bound or type$(x_j) = 3$, $d_j(t)$ remains feasible only if $t = \dfrac{d_j(0)}{\alpha_{pj}} = 0$. For $t > 0$ it becomes infeasible and $j$ joins $M$.

(b) $\alpha_{pj} < 0$, $d_j(t)$ is increasing

i. $d_j(0) < 0$

A. If $d_j(0)$ is infeasible, i.e., $j \in M$, $d_j(t)$ remains infeasible as long as $t < \dfrac{d_j(0)}{\alpha_{pj}}$. If type $(x_j) = 3$, it becomes infeasible again when $t > \dfrac{d_j(0)}{\alpha_{pj}}$ and $j$ joins $P$.

B. If $d_j(0)$ is feasible and $x_j$ is a bounded (type-1) variable then $d_j(t)$ remains feasible as long as $t \leq \dfrac{d_j(0)}{\alpha_{pj}}$, after which it becomes positive and $j$ joins $P$.

ii. If $d_j(0) = 0$ and, additionally, $x_j$ is at upper bound or type$(x_j) = 3$ then it remains feasible only for $t = \dfrac{d_j(0)}{\alpha_{pj}} = 0$; for $t > 0$ it becomes positive and $j$ joins $P$.

**II.** $t \leq 0$, i.e., the outgoing variable leaves at zero.

(a) $\alpha_{pj} > 0$, i.e., $d_j(t)$ is increasing

i. $d_j(0) < 0$

A. If $d_j(0)$ is infeasible, i.e., $j \in M$, $d_j(t)$ remains infeasible as long as $t > \dfrac{d_j(0)}{\alpha_{pj}}$. If type$(x_j) = 3$, it becomes infeasible again when $t < \dfrac{d_j(0)}{\alpha_{pj}}$ and $j$ joins $P$.

B. If $d_j(0)$ is feasible, i.e., $x_j$ is at upper bound, then $d_j(t)$ remains feasible as long as $t \geq \dfrac{d_j(0)}{\alpha_{pj}}$, after which it becomes positive and $j$ joins $P$.

ii. If $d_j(0) = 0$ and, additionally, $x_j$ is at upper bound or type$(x_j) = 3$ then $d_j(t)$ remains feasible only for $t = \dfrac{d_j(0)}{\alpha_{pj}} = 0$; for $t < 0$ it becomes positive and $j$ joins $P$.

(b) $\alpha_{pj} < 0$, i.e., $d_j(t)$ is decreasing

i. $d_j(0) > 0$

A. If $d_j(0)$ is infeasible, $j \in P$, type$(x_j) = 3$, $d_j(t)$ remains infeasible as long as $t > \dfrac{d_j(0)}{\alpha_{pj}}$ and it becomes infeasible again if $t < \dfrac{d_j(0)}{\alpha_{pj}}$ when $j$ joins $M$.

B. If $d_j(0)$ is feasible ($x_j$ is at upper bound), $d_j(t)$ remains feasible as long as $t \geq \dfrac{d_j(0)}{\alpha_{pj}}$ after which it becomes negative and $j$ joins $M$.

ii. $d_j(0) = 0$, and, additionally, $x_j$ is at lower bound or type$(x_j) = 3$ then $d_j(t)$ remains feasible only if $t = \dfrac{d_j(0)}{\alpha_{pj}} = 0$; for $t < 0$ it becomes negative and $j$ joins $M$.

The above discussion can be summarized as follows.

1. If $t \geq 0$ is required then the dual feasibility status of $d_j$ (and set $M$ or $P$, thus the composition of $f(t)$) changes for values of $t$ defined by positions where
   $d_j < 0$ and $\alpha_{pj} < 0$ or
   $d_j \geq 0$ and $\alpha_{pj} > 0$

2. If $t \leq 0$ is required then the critical values are defined by
   $d_j < 0$ and $\alpha_{pj} > 0$ or
   $d_j \geq 0$ and $\alpha_{pj} < 0$.

The second case can directly be obtained from the first one by using $-\alpha_{pj}$ in place of $\alpha_{pj}$. In both cases there are some further possibilities. Namely, if type$(x_j) = 3$ (free variable) and $d_j \neq 0$ then at the critical point the feasibility status of $d_j$ changes twice. First when it becomes zero (feasible), and second, when it becomes nonzero again. In both cases the critical value is $d_j/\alpha_{pj}$. It is also worth keeping track of changes of dual feasibility of upper bounded (type-1) positions. Though they will be treated by 'feasibility correction' the required bound swaps have to be recorded. The critical values of $t$ (for the $t \geq 0$ case) are defined by $d_j/\alpha_{pj}$ if $x_j$ is at lower bound and $\alpha_{pj} > 0$ or if $x_j$ is at upper bound and $\alpha_{pj} < 0$. If the finally chosen value of $t$ is larger than some of the threshold values defined

by bounded variables then these variables will take part in bound swap for feasibility correction.

Let the critical values defined above for $t \geq 0$ be arranged in an ascending order: $0 \leq t_1 \leq \cdots \leq t_Q$, where $Q$ denotes the total number of them. For $t \leq 0$ we make a reverse ordering: $t_Q \leq \cdots \leq t_1 \leq 0$, or equivalently, $0 \leq -t_1 \leq \cdots \leq -t_Q$. Now we are ready to investigate how $f(t)$ characterizes the change of dual infeasibility.

Clearly, $Q$ cannot be zero, i.e., if row $p$ has been selected as a candidate it defines at least one critical value, see (8). Assuming $v_p < 0$ the initial slope of $f(t)$, according to (7), is

$$s_p^0 = -v_p = \sum_{j \in P} \alpha_{pj} - \sum_{j \in M} \alpha_{pj}. \tag{13}$$

Now $t \geq 0$ is required, so we try to move away from $t = 0$ in the positive direction. $f(t)$ keeps improving at the rate of $s_p^0$ until $t_1$. At this point $d_{j_1}(t_1) = 0$, $j_1$ denoting the position that defined the smallest ratio $t_1 = \dfrac{d_{j_1}(0)}{\alpha_{pj_1}}$. At $t_1$ the feasibility status of $d_{j_1}$ changes. Either it becomes feasible at this point or it becomes infeasible after $t_1$.

If $t_1 \geq 0$ then either (a) $d_{j_1} \geq 0$ and $\alpha_{pj_1} > 0$ or (b) $d_{j_1} \leq 0$ and $\alpha_{pj_1} < 0$. In these cases:

(a) $d_{j_1}(t)$ is decreasing.

  (i) If $d_{j_1}$ was feasible it becomes infeasible and $j_1$ joins $M$. At his point $s_p^0$ decreases by $\alpha_{pj_1}$, see (13).

  (ii) If $d_{j_1}$ was infeasible ($j_1 \in P$) it becomes feasible and $j_1$ leaves $P$. Consequently, $s_p^0$ decreases by $\alpha_{pj_1}$.

  If $d_{j_1} = 0$ then we only have (i).

(b) $d_{j_1}(t)$ is increasing.

  (i) If $d_{j_1}$ was feasible it becomes infeasible and $j_1$ joins $P$. At his point $s_p^0$ decreases by $-\alpha_{pj_1}$, see (13).

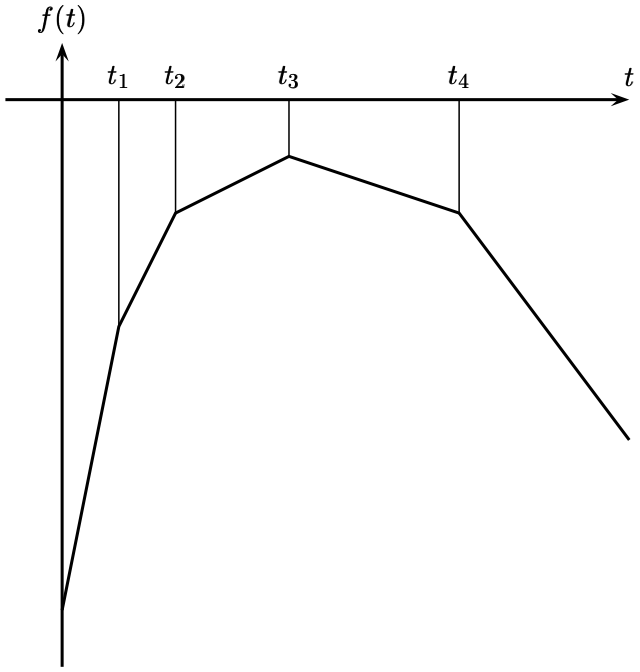


Figure 2: The sum of dual infeasibilities as a function of $t$.

(ii) If $d_{j_1}$ was infeasible ($j_1 \in M$) it becomes feasible and $j_1$ leaves $M$. Consequently, $s_p^0$ decreases by $-\alpha_{pj_1}$.

If $d_{j_1} = 0$ then we only have (i).

Cases (a) and (b) can be summarized by saying that at $t_1$ the slope of $f(t)$ decreases by $|\alpha_{pj_1}|$ giving $s_p^1 = s_p^0 - |\alpha_{pj_1}|$. If $s_p^1$ is still positive we carry on with the next point ($t_2$), and so on. The above analysis is valid at each point. Clearly, $f(t)$ is linear between two neighboring threshold values. For obvious reasons, these values are called *breakpoints*. The distance between two points can be zero if a breakpoint has a multiplicity $> 1$. Since the slope decreases at breakpoints $f(t)$ *is a piecewise linear concave function* as illustrated in Figure 2. It achieves its maximum when the slope changes sign. This is a global maximum. After this point the dual objective starts deteriorating.

If $v_p > 0$ then $t \leq 0$ is required. In this case the above analysis remains valid if $\alpha_{pj}$ is

substituted by $-\alpha_{pj}$. It is easy to see that both cases are covered if we take $s_p^0 = |v_p|$ and

$$s_p^k = s_p^{k-1} - |\alpha_{pj_k}|, \text{ for } k = 1, \ldots, Q.$$

## 3.2 Dual phase-1 step with all types of variables

Let $t_0 = 0$ and $f_i = f(t_i)$. Obviously, the sum of dual infeasibilities in the breakpoints can be computed recursively as $f_i = f_{i-1} + s_p^{i-1}(t_i - t_{i-1})$, for $i = 1, \ldots, Q$.

Below, we give the description of one iteration of the algorithm which we call GDPO (for Generalized Dual Phase One). We assume that dual feasibility correction according to (2) has been carried out before the iterations started. During ratio test, GDPO may introduce breakpoints for upper bounded variables. The presented algorithm properly handles these 'break points'. Namely, the passed ones are recorded and will take part in feasibility correction.

**An iteration of the Generalized Dual Phase-1 (GDPO) algorithm:**

1. Identify sets $M$ and $P$ as defined in (1) and (4). If both are empty, solution is dual feasible, procedure terminates.
2. Form auxiliary vector $\tilde{a} = \sum_{j \in M} a_j - \sum_{j \in P} a_j$.
3. Compute the vector of dual phase-1 reduced costs: $v = B^{-1}\tilde{a}$, as in (11).
4. Select an improving candidate row according to some rule (i.e., Dantzig [2] or Devex [6, 3]), denote its basic position by $p$. This will be the pivot row.

   If none exists, terminate: The dual problem is infeasible.
5. Compute the $p$-th row of $B^{-1}$: $\beta^T = e_p^T B^{-1}$ and determine nonbasic components of the updated pivot row by $\alpha_{pj} = \beta^T a_j$ for $j \in N$.
6. Compute dual ratios for eligible positions according to rules under **I.**, if $v_p < 0$, or **II.**, if $v_p > 0$, as discussed in section 3.1. Store their absolute values in a sorted order: $0 \leq |t_1| \leq \cdots \leq |t_Q|$.

7. Set $i = k = 0,\ t_0 = 0,\ f_0 = f(0),\ \ s_p^0 = |v_p|$ and $T^+ = T^- = \emptyset$; $k$ will run through all breakpoints defined in Step 6, $i$ covers those which do not define bound swaps.

   **While** $k < Q$ and $s_p^k \geq 0$ **do**

   $k := k + 1$

   Let $j_k$ denote the column index of the variable that defined the $k$-th smallest ratio, $|t_k|$.

   **If** type$(x_{j_k}) = 1$ move $x_{j_k}$ to its opposite bound, i.e., set $x_{j_k} = u_{j_k}$ or $x_{j_k} = 0$, and $T^+ = T^+ \cup \{j_k\}$ or $T^- = T^- \cup \{j_k\}$. Bound swaps do not change the slope of $f$.

   **else** Compute $f_k = f_i + s_p^i(t_k - t_i),\ s_p^k = s_p^i - |\alpha_{pj_k}|,\ i = k$.

   **end while**

   Let $q$ denote the index of the last breakpoint for which the slope $s_p^k$ was still nonnegative, $q = j_k$. The maximum of $f(t)$ is achieved at this break point. The incoming variable is $x_q$.

8. Update solution:

   (a) Take care of basis change:

   Compute $\alpha_q = B^{-1}a_q$

   Update $x_B$: $\bar{x}_B = Ex_B$, $E$ denoting the elementary transformation matrix created from $\alpha_q$.

   Update basis inverse ($\bar{B}^{-1} = EB^{-1}$) and the basic/nonbasic index sets.

   (b) Take care of bound swaps in the new basis:

$$\bar{x}_B := \bar{x}_B - \sum_{j \in T^+} u_j \bar{\alpha}_j + \sum_{j \in T^-} u_j \bar{\alpha}_j, \tag{14}$$

   where the sum is defined to be 0 if the corresponding index set is empty. Also, the list of variables at upper bound has to be updated. Note: $\bar{\alpha}_j$s in (14) are defined in the new basis: $\bar{\alpha}_j = \bar{B}^{-1}a_j$.

### 3.2.1 Work per iteration

It is easy to see that the extra work required for GDPO is generally small.

1. Ratio test: same work as with traditional dual.

2. The break points of the piecewise linear dual objective function have to be stored and sorted. This requires extra memory for the storage, and extra work to sort. However, the $t_k$ values have to be sorted only up to the point where $f(t)$ reaches its maximum. Therefore, if an appropriate priority queue is set up for these values the extra work can be kept at minimum.

3. Taking care of bound swaps according to (3) requires the transformation (FTRAN) of a single composite column.

### 3.2.2 Implementation

For the efficient implementation of GDPO a sophisticated data structure (priority queue) is needed to store and (partially) sort the break points. Additionally, since row and column operations are performed on $A$, it is important to have a data structure that supports efficient row and columnwise access of $A$.

### 3.2.3 Key features of GDPO

The main computational advantage of the introduced GDPO algorithm that it can make multiple steps with one updated pivot row. These steps correspond to several traditional iterations. A multiple step of GDPO requires very little extra work compared to the traditional method.

GDPO is an efficient generalization of the traditional dual simplex algorithm in the sense that the latter stops at the smallest ratio (first breakpoint) while GDPO can pass many breakpoints making the maximum progress towards dual feasibility with the selected outgoing variable. Its efficiency of is not hampered by the presence of all types of variables.

GDPO possesses a favorable anti-degeneracy property. It can pass zero ratios in two different ways: (i) the $|\alpha_{pj_k}|$ of a zero ratio does not make the slope of $f$ change sign, (ii) $d_{j_k} = 0$ for a type-1 variable and $\alpha_{pj_k}$ has the 'wrong' sign ($d_j$ is moving in the infeasible domain). This ratio is simply ignored and a bound swap is triggered for the corresponding $x_{j_k}$.

The freedom of multiple choice created by the break points can be used to enhance the numerical stability of GDPO. Namely, if $|\alpha_{pj_q}|$ of the optimal breakpoint is too small and $q > 1$ we can take one of the previous breakpoints with sufficiently large pivot element.

# 4 Two examples of the algorithmic step

The operation of GDPO is demonstrated on two examples. We assume pivot row $p$ has been selected based on $v_p$ and the updated pivot row has been determined. The types of the variables and the $d_j$ values are given. The status of upper bounded variables is also indicated (LB for 'at lower bound' and UB for 'at upper bound').

**Example-1.** The problem has 12 nonbasic variables. Now $v_p = 10$ which is the $t \leq 0$ case. The sum of dual infeasibilities is $f = -35$. The solution is dual degenerate.

| $j$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| type($x_j$) | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | |
| Status | | | | | | | | | $LB$ | $LB$ | $UB$ | $UB$ | |
| $\alpha_{pj}$ | 8 | 4 | 1 | 4 | 2 | −1 | −1 | 1 | −1 | 1 | −1 | 1 | $v_p = 10$ |
| $d_j$ | −24 | 2 | 0 | −8 | −1 | 1 | 0 | 0 | 0 | 1 | −1 | −2 | $f = -35$ |
| Infeasibility | $M$ | $P$ | | $M$ | $M$ | | | | | | | | |
| Ratio | −3 | | 0 | −2 | −0.5 | −1 | 0 | | 0 | | | −2 | |
| 2nd ratio | −3 | | | | | | | | | | | | |

Altogether, 9 ratios have been defined, $Q = 9$. Note, for the first position there are two identical ratios. After sorting the absolute values of the ratios:

| Index | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| $j_k$ | 9 | 7 | 3 | 5 | 6 | 4 | 12 | 1 | 1 |
| $\lvert t_{j_k}\rvert$ | 0 | 0 | 0 | 0.5 | 1 | 2 | 2 | 3 | 3 |
| $\alpha_{pj_k}$ | 1 | −1 | 1 | 2 | 1 | 4 | 1 | 8 | 8 |

Now, applying Step 7 of GDPO, we obtain

| $k$ | $i$ | $j_k$ | $\lvert t_k\rvert$ | $\alpha_{pj_k}$ | $f_k = f_i + s_p^i(t_k - t_i)$ | $s_p^k = s_p^i - \lvert\alpha_{pj_k}\rvert$ | Remarks |
|---|---|---|---|---|---|---|---|
| 0 | 0 | | 0 | | $-35$ | 10 | |
| 1 | 0 | 9 | 0 | 1 | $-35$ | 10 | BSW for $x_9$ |
| 2 | 0 | 7 | 0 | $-1$ | $-35$ | $10 - \lvert-1\rvert = 9$ | |
| 3 | 2 | 3 | 0 | 1 | $-35$ | $9 - \lvert1\rvert = 8$ | |
| 4 | 3 | 5 | 0.5 | 2 | $-35 + 8 \times (0.5 - 0) = -31$ | $8 - \lvert2\rvert = 6$ | |
| 5 | 4 | 6 | 1 | 1 | $-31 + 6 \times (1 - 0.5) = -28$ | $6 - \lvert1\rvert = 5$ | |
| 6 | 5 | 4 | 2 | 4 | $-28 + 5 \times (2 - 1) = -23$ | $5 - \lvert4\rvert = 1$ | |
| 7 | 6 | 12 | 2 | 1 | $-23$ | 1 | BSW for $x_{12}$ |
| 8 | 6 | 1 | 3 | 8 | $-23 + 1 \times (3 - 2) = -22$ | $1 - \lvert8\rvert = -7$ | |

BSW stands for bound swap. We have used 8 of the 9 breakpoints. At termination of this step of GDPO $k = 8$, therefore, the entering variable is $x_1$ that has defined $t_8$. The dual steplength is $-3$ $(= t_8)$. Traditional methods would have stopped at the first breakpoint resulting in a degenerate iteration.

It is instructive to monitor how dual infeasibility changes at the nonzero breakpoints.

| $j$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| type($x_j$) | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | |
| $\alpha_{pj}$ | 8 | 4 | 1 | 4 | 2 | −1 | −1 | 1 | −1 | 1 | −1 | 1 | |
| $d_j$ | −24 | 2 | 0 | −8 | −1 | 1 | 0 | 0 | 0 | 1 | −1 | −2 | $f = -35$ |
| Infeasibility | $M$ | $P$ | | $M$ | $M$ | | | | | | | | |
| $d_j + 0.5\alpha_{pj}$ | −20 | 4 | 0.5 | −6 | 0 | 0.5 | −0.5 | 0.5 | −0.5 | 1.5 | −1.5 | −1.5 | $f = -31$ |
| Infeasibility | $M$ | $P$ | $P$ | $M$ | | | $M$ | | $(M)$ | | | | |
| $d_j + \alpha_{pj}$ | −16 | 6 | 1 | −4 | 1 | 0 | −1 | 1 | −1 | 2 | −2 | −1 | $f = -28$ |
| Infeasibility | $M$ | $P$ | $P$ | $M$ | | | $M$ | | $(M)$ | | | | |
| $d_j + 2\alpha_{pj}$ | −8 | 10 | 2 | 0 | 3 | −1 | −2 | 2 | −2 | 3 | −3 | 0 | $f = -23$ |
| Infeasibility | $M$ | $P$ | $P$ | | | $M$ | $M$ | | $(M)$ | | | | |
| $d_j + 3\alpha_{pj}$ | 0 | 14 | 3 | 4 | 5 | −2 | −3 | 4 | −3 | 4 | −4 | 1 | $f = -22$ |
| Infeasibility | | $P$ | $P$ | | | $M$ | $M$ | | $(M)$ | | | $(P)$ | |

$(M)$ and $(P)$ denote temporarily infeasible positions which are subject to feasibility correction by bound swap. Such positions do not contribute to $f$. At the end of Step 7 of GDPO we still have four infeasible positions, $P = \{2, 3\},\ M = \{6, 7\}$. Some infeasibilities disappeared and new ones were created but the sum of infeasibilities has been reduced from $-35$ to $-22$. It is interesting to see what happens if we increase $t$ to 4. The outcome is $f = -37$ which shows a rate of deterioration of 15. The reason for this rate is that at $t = 3$ we have a breakpoint with multiplicity of 2 (see line '2nd ratio'), both defined by $x_1$. Therefore, passing through it, the slope decreases by $2|\alpha_{p1}| = 16$ from $+1$ to $-15$.

**Example-2.** This problem has 5 nonbasic variables. Here, $v_p = -4$ which is the $t \geq 0$ case. The sum of dual infeasibilities is $f = -8$.

| $j$ | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| type($x_j$) | 2 | 1 | 1 | 2 | 3 | |
| Status | | $UB$ | $LB$ | | | |
| $\alpha_{pj}$ | $-2$ | $-3$ | 2 | $-1$ | 1 | $v_p = -4$ |
| $d_j$ | $-4$ | 0 | 1 | $-1$ | 3 | $f = -8$ |
| Infeasibility | $M$ | | | $M$ | $P$ | |
| Ratio | 2 | 0 | 0.5 | 1 | 3 | |

Altogether, 5 ratios have been defined, $Q = 5$. After sorting (the absolute values of) the ratios:

| Index | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $j_k$ | 2 | 3 | 4 | 1 | 5 |
| $\lvert t_{j_k} \rvert$ | 0 | 0.5 | 1 | 2 | 3 |
| $\alpha_{pj_k}$ | $-1$ | 2 | $-3$ | $-2$ | 1 |

Applying Step 7 of GDPO, we obtain

| $k$ | $i$ | $j_k$ | $\lvert t_k \rvert$ | $\alpha_{pj_k}$ | $f_k = f_i + s_p^i(t_k - t_i)$ | $s_p^k = s_p^i - \lvert \alpha_{pj_k} \rvert$ | Remarks |
|---|---|---|---|---|---|---|---|
| 0 | 0 | | 0 | | $-8$ | 4 | |
| 1 | 0 | 2 | 0 | $-3$ | $-8$ | 4 | BSW for $x_2$ |
| 2 | 0 | 3 | 0.5 | 2 | $-8$ | 4 | BSW for $x_3$ |
| 3 | 0 | 4 | 1 | $-1$ | $-8 + 4 \times (1 - 0) = -4$ | $4 - \lvert -1 \rvert = 3$ | |
| 4 | 3 | 1 | 2 | $-2$ | $-4 + 3 \times (2 - 1) = -1$ | $3 - \lvert -2 \rvert = 1$ | |
| 5 | 4 | 5 | 3 | 1 | $-1 + 1 \times (3 - 2) = 0$ | $1 - \lvert 1 \rvert = 0$ | |

Now, we have used up all 5 breakpoints. Two of them will be handled by feasibility correction. The other three have contributed to the reduction of dual infeasibilities. The last breakpoint defines dual steplength of 3 ($= t_5$). and the entering variable $x_{j_5} = x_5$. In this case GDPO could reach dual feasibility with one updated pivot row in one

step. Traditional methods would have stopped (at best) at the first non-BSW breakpoint resulting in an improved but still dual infeasible solution.

Again, we can trace the change of dual infeasibilities at the genuine breakpoints (other than bound swaps).

| | | | | | | |
|---|---|---|---|---|---|---|
| $j$ | 1 | 2 | 3 | 4 | 5 | |
| type($x_j$) | 2 | 1 | 1 | 2 | 3 | |
| $\alpha_{pj}$ | $-2$ | $-3$ | 2 | $-1$ | 1 | |
| $d_j$ | $-4$ | 0 | 1 | $-1$ | 3 | $f = -8$ |
| Infeasibility | $M$ | | | $M$ | $P$ | |
| $d_j - \alpha_{pj}$ | $-2$ | 3 | $-1$ | 0 | 2 | $f = -4$ |
| Infeasibility | $M$ | $(P)$ | $(M)$ | | $P$ | |
| $d_j - 2\alpha_{pj}$ | 0 | 6 | $-3$ | 1 | 1 | $f = -1$ |
| Infeasibility | | $(P)$ | $(M)$ | | | |
| $d_j - 3\alpha_{pj}$ | 2 | 9 | $-5$ | 2 | 0 | $f = 0$ |
| Infeasibility | | $(P)$ | $(M)$ | | | |

It is important to emphasize that tracing infeasibility as shown above is actually not done in GDPO as there is no need for that. We included it only to demonstrate the outcome of different possible choices of the incoming variable.

# 5 Summary

We have presented a generalization of the dual phase-1 algorithms that handles all types of variables efficiently. It is based on the piecewise linear nature of the dual objective if defined as a function of releasing one basic equation. The main advantage is that a number of very cheap iterations can be made with one updated pivot row. As additional benefit, GDPO possesses several favorable features making it particularly suitable for inclusion in optimization software.

We have shown GDPO can be implemented efficiently. The given examples have clearly

indicated its superiority over the traditional method that stops at the first breakpoint. Similar pattern can be expected in case of large scale problems resulting in huge savings in overall computational work in dual phase-1.

As a last point, we indicate that for the selection of the pivot row any of the known methods can be used, including the Dantzig rule [2], dual Devex [6] or dual steepest edge [3]. In practice the combination of dual steepest edge and GDPO proved to be particularly efficient.

# References

[1] Chvátal, V., *Linear Programming*, Freeman and Co., 1983.

[2] Dantzig, G.B., *Linear Programming and Extensions*, Princeton University Press, Princeton, N.J., 1963.

[3] Forrest, J.J., Goldfarb, D., "Steepest edge simplex algorithms for linear programming" *Mathematical Programming*, 57, 1992, No. 3., p. 341–374.

[4] Fourer, R., "Notes on the Dual Simplex Method", Unpublished, March, 1994.

[5] Greenberg, H.J., "Pivot selection tactics", in Greenberg, H.J. (ed.), *Design and Implementation of Optimization Software*, Sijthoff and Nordhoff, 1978, p. 109–143.

[6] Harris, P.M.J., "Pivot Selection Method of the Devex LP Code", *Mathematical Programming*, 5, 1973, p. 1–28.

[7] Lemke, C.E., "The Dual Method of Solving the Linear Programming Problem", *Naval Research Logistics Quarterly*, 1, 1954, p. 36–47.

[8] Maros, I., "A general Phase-I method in linear programming", *European Journal of Operational Research*, 23(1986), p. 64–77.

[9] Maros, I., "A Piecewise Linear Dual Procedure in Mixed Integer Programming", in F. Giannesi, R. Schaible, S. Komlosi (eds.), *New Trends in Mathematical Programming*, Kluwer Academic Publishers, 1998, pp. 159–170.

[10] Maros, I., Mitra, G., "Simplex Algorithms", Chapter 1 in Beasley J. (ed.) *Advances in Linear and Integer Programming*, Oxford University Press 1996, p. 1–46.

[11] Orchard-Hays, W., *Advanced Linear-Programming Computing Techniques*, McGraw-Hill, 1968.

[12] Padberg, M., "Linear Optimization and Extensions", Springer, 1995.

[13] Wolfe, Ph., "The composite simplex algorithm", *SIAM Review*, 7 (1), 1965, p. 42–54.